Linear Regression

Contents

Introduction

Linear regression is an essential statistical tool used to model the relationship between a dependent variable and one or more independent variables. The simplest case is the simple linear regression model, where there is only one independent variable. Formally, we consider n observations (x_i, y_i), where y_i is a random observation of the variable Y_i, with an expectation $\mu_i = E(Y_i)$. In its most common form, the model is expressed as:

$$Y_i = \mu_i + \varepsilon_i, \text{ where } \mu_i = x_i\beta$$

where β is an unknown parameter to be estimated and ε_i is an error term, assumed to be random with a mean of zero and a constant variance. Additionally, the assumption of normality is made in order to make inferences about the estimated coefficients, thus $\varepsilon_i \sim N(0, \sigma^2)$ for $i = 1, \ldots, n$.

In the case of having p explanatory variables, the multiple linear regression model is considered:

$$\mu_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}$$

where $\beta_0, \beta_1, \ldots, \beta_p$ are the coefficients to be estimated. To obtain these estimates, n equations are set up with the available data from a sample:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \varepsilon_i, \text{ for } i = 1, ..., n. \tag{1}$$

Let $\mathbf{X}$ be the data matrix containing the values of the independent variables, and β is the vector of coefficients to be estimated. The matrix $\mathbf{X}$ typically includes a column of ones to account for the intercept term β_0.

Estimation of Model Coefficients

The estimation of the model coefficients is commonly performed using the method of ordinary least squares (OLS). This method seeks to minimize the sum of the squares of the residuals, which are the differences between the observed values of y and the values predicted by the model,

$$\text{Min } ||y - \mathbf{X}\beta||^2. \tag{2}$$

The least squares estimator is

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T y.$$

Simulating data

First, we generate simulated data for our linear regression model. We create the independent variable x from a uniform distribution and we generate random errors. Then, we can compute the observed and true values of the dependent variable y.

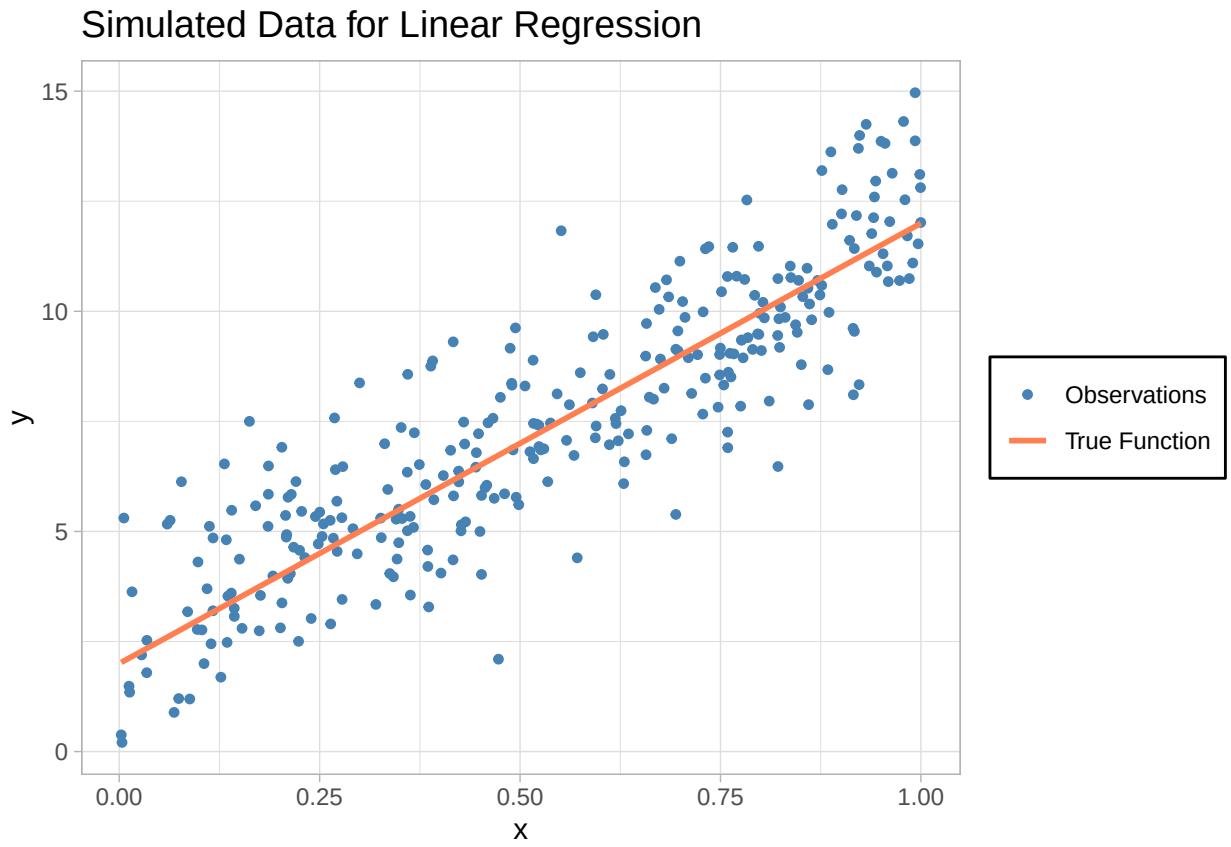
```
x <- runif(300, min=0, max=1) # X ~ Unif(0,1) n=300
x <- x[order(x)]

# Errors
eps <- rnorm(300, sd = 1.5)
# True and observed values
y.true <- 2 + 10*x
y.obs <- y.true + eps
```

To visualize the data, we utilize `ggplot2` and `gridExtra`:

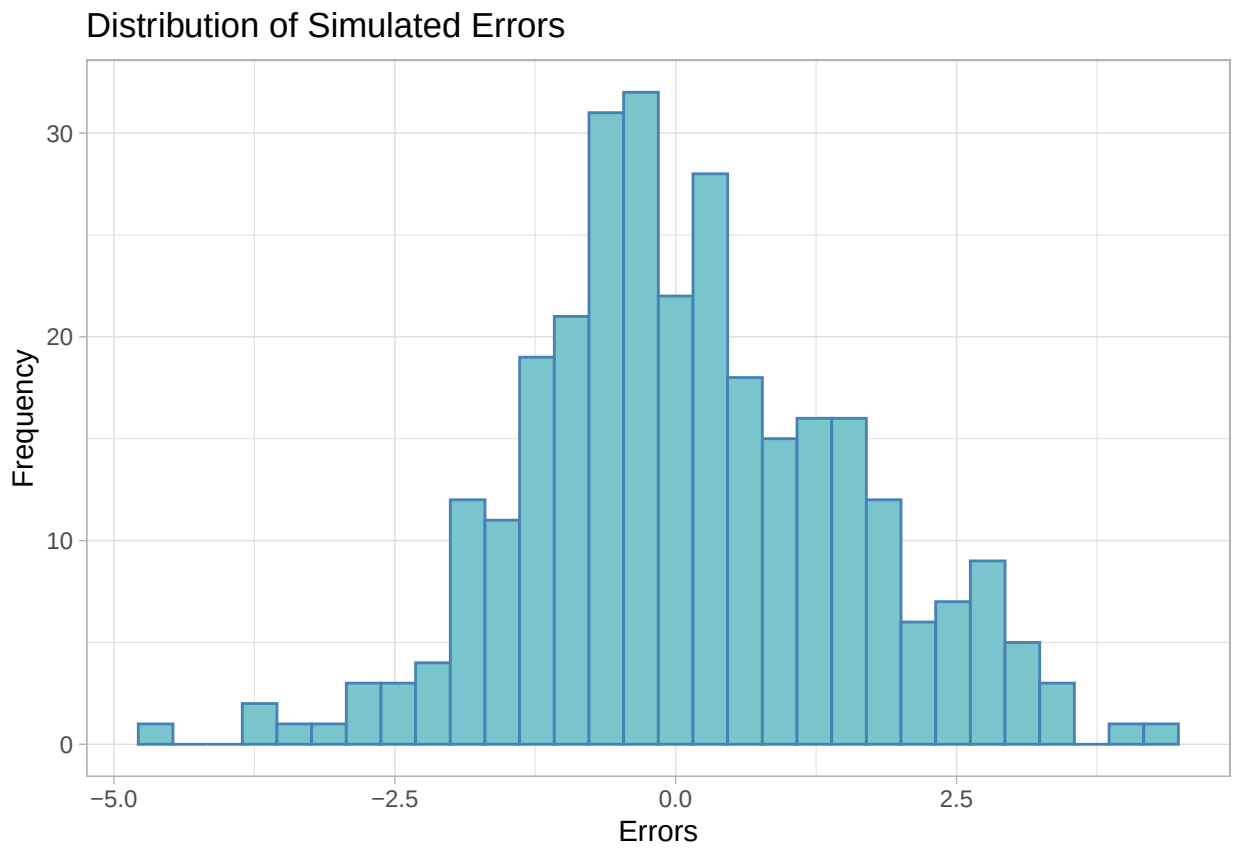
```
library(ggplot2)
library(gridExtra)
```

```
data <- data.frame(x, y.obs, y.true)
ggplot(data = data) +
  geom_point(aes(x = x, y = y.obs, color = "Observations"), shape = 20, size = 2) +
  geom_line(aes(x = x, y = y.true, color = "True Function"), linewidth = 1) +
  scale_color_manual(values = c("Observations" = "steelblue", "True Function" = "coral")) +
  labs(title = "Simulated Data for Linear Regression",
       x = "x",
       y = "y",
       color = NULL) +
  theme_light() +
  theme(plot.title = element_text(size = 14),
        legend.position.inside = c(0.15, 0.85),
        legend.background = element_rect(fill = "white", colour = "black"))
```



Additionally, we visualize the distribution of the simulated errors.

```
ggplot(data = data.frame(eps), aes(x = eps)) +
  geom_histogram(bins = 30, color = 'steelblue', fill = "cadetblue3") +
  labs(title = "Distribution of Simulated Errors", x = "Errors", y = "Frequency") +
  theme_light()
```



Fitting the Linear Regression Model

Now, we fit a linear regression model to the observed data using the `lm` function in R.

```
model <- lm(y.obs ~ x, data = data)
model$coefficients
```

```
## (Intercept)           x
##    2.355573    9.555866
```

```
x.pred <- seq(0, 1, length.out = 300)
y.pred <- predict(model, newdata = data.frame(x = x.pred))
```

We visualize the fitted model along with the observed data and the true function.

```
data.model <- data.frame(x, y.obs, y.true, x.pred, y.pred)
ggplot(data = data.model) +
  geom_point(aes(x = x, y = y.obs, color = "Observations"), shape = 20, size = 2) +
  geom_line(aes(x = x, y = y.true, color = "True Function"), linewidth = 1) +
  geom_path(aes(x = x.pred, y = y.pred, color = "Prediction"), linewidth = 1) +
  scale_color_manual(values = c("Observations" = "steelblue",
                                "Prediction" = "coral",
                                "True Function" = "palegreen3")) +
  labs(title = "Linear Regression",
       x = "x",
       y = "y",
       color = NULL) +
```

```
  theme_light() +
  theme(plot.title = element_text(size = 14),
        legend.position.inside = c(0.18, 0.85),
        legend.background = element_rect(fill = "white", colour = "gray50"))
```

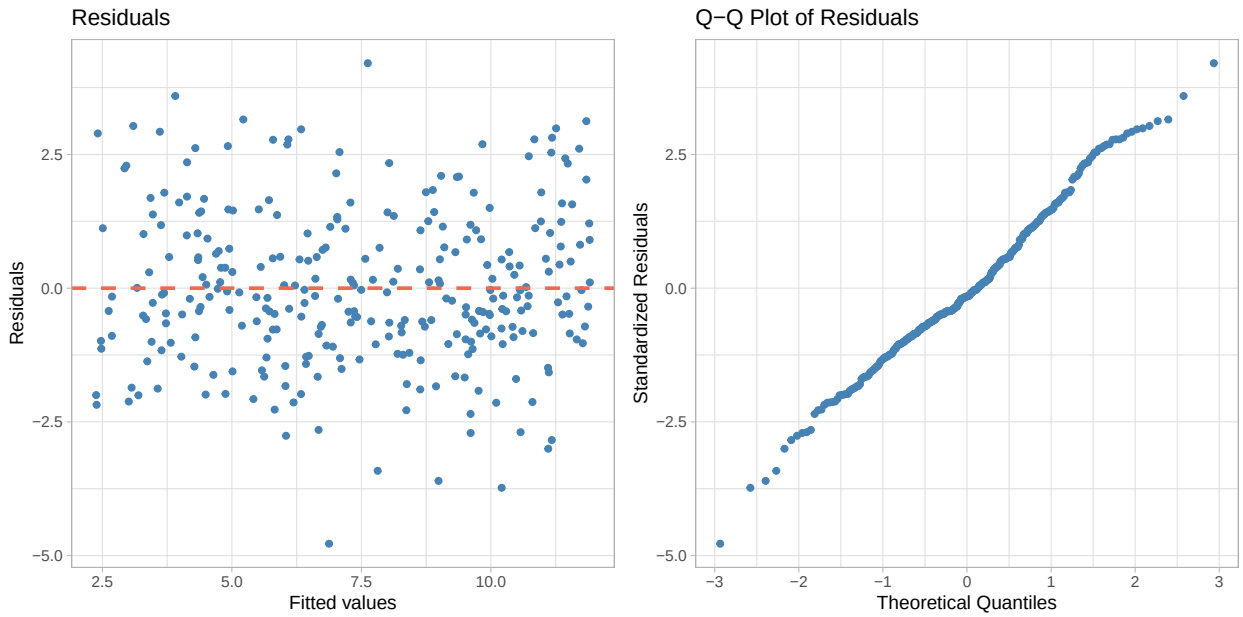
Finally, we perform a residual analysis to check the goodness of fit of our model. We create plots for the residuals versus fitted values and a Q-Q plot to assess the normality of the residuals.

```
residuals <- resid(model)
# Residuals plot
plot1 <- ggplot(data, aes(x = fitted(model), y = residuals)) +
  geom_point(color = "steelblue") +
  geom_hline(yintercept = 0, linetype = "dashed", color = "coral2", linewidth=1) +
  labs(title = "Residuals",
       x = "Fitted values",
       y = "Residuals",
       color = NULL) +
  theme_light()

# Q-Q plot
plot2 <- ggplot() +
  stat_qq(aes(sample = residuals), color = "steelblue") +
  labs(title = "Q-Q Plot of Residuals",
       x = "Theoretical Quantiles",
       y = "Standardized Residuals",
       color = NULL) +
```

```
  theme_light()

# Arrange plots side by side
plot <- grid.arrange(plot1, plot2, nrow = 1, ncol = 2)
```



Manual Calculation of $\hat{\beta}$

To manually calculate the coefficients $\hat{\beta}$ using the formula $\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T y$, we do:

```
# Create the design matrix X
X <- cbind(1, x)  # Add a column of ones for the intercept term

# Calculate X^T * X
XtX <- t(X) %*% X

# Calculate (X^T * X)^-1
XtX_inv <- solve(XtX)

# Calculate X^T * y
Xty <- t(X) %*% y.obs

# Calculate beta
beta_hat <- XtX_inv %*% Xty
beta_hat
```

```
##       [,1]
##   2.355573
## x 9.555866
```